Applying Diversity Metrics to Improve the Selection of Web Search Term Refinements

Tague Griffith
tague@minion.net

Jan Pfeifer
jan.pfeifer@yahoo.com.br

ABSTRACT

Automatic expansion of user queries during web search is a technique provided by most web search engines. Term expansion helps users to phrase more specific queries, correct misspellings, and acts as an accelerator for frequently used queries. Typically search engines provide five to ten refinements. Many of the refinement sets have a high degree of redundancy which suggests that popularity is the primary factor for selecting a refinement. This is particularly evident in terms that have multiple semantic meanings one of which is association with technology. The popularity of the technological usage of the word with dwarf the other usages, monopolizing the set of refinements. We explore two approaches to apply a measure of diversity in the selection of refinements: using word sense similarity and web semantic similarity. We analyze some of the problems with using diversity as the sole metric for refinement selection and evaluate the results of our technique through a human editorial test.

1. INTRODUCTION

Search term refinements are suggestions offered to a user by a search engine to help formulate a more specific query. As an example, if you search for *java* on Yahoo! the search engine will suggest *java downloads*, *java games*, *mobile java general*, *free java download*, and *java runtime environment* as follow on queries to find the data you are interested in. (You can also see this feature by typing a search term into the Firefox search box for most of the major search engines.) All of the major search engines implement some form of term refinement.

If we look at the Google Suggest refinements for the term *Amazon* in Table 1, we notice that all of the potential query refinements are related to the company Amazon. The results are tightly clustered around a single concept despite the fact that *Amazon river* has 4,000,000 results [1] - over 5 times as many as *Amazon Kindle*.

This is a pattern we see repeated over and over to varying degrees in all web search engines. The refinements for terms with several distinct clusters, particularly when one of those senses is technology related, are skewed toward a single senses cluster.

For many applications it makes sense to drive the refinement terms using a model of query popularity, but as we see in features like spelling correction, popularity is not always the best metric for every application. If you are interested in presenting an exploratory interface to the web, a different technique for selecting search term refinements - one that chooses a diverse sense of meaning, might make a better choice.

Diversity can be a difficult concept to apply across a large population. Continuing on with our Amazon example, *Amazon river* makes sense as a refinement with broad applicability to wide class of the population but using *Amazon Women in the Mood* as a refinement [2] would only be applicable to users interested in the television series *Futurama*. *Amazon Women in the Mood* would certainly increase the diversity of the refinement set, but since only a small number of refinements can be displayed to the end user, is it widely applicable enough to warrant displaying it.

Looking at some sample suggestions, one immediately notices that some of the redundancy could have been easily identified by simple lexical means: stemming, stop words removal, spelling corrections, but much of the conceptual redundancy requires deeper analysis.

In this paper we propose a mechanism for selecting search term refinements based on a linear combination of a diveristy and popularity score. We compare the performance of two different metrics of diversity, one semantic and one information retrieval, to see their overall performance.

2. RELATED WORK

In the field of Information Retrieval, many different techniques for automatic query expansion have been investigated. One of the classic techniques for query expansion is

Suggestion	Result Pages
amazon uk	13,100,000
amazon kindle	783,000
amazon mp3	12,500,000
amazon.fr	62,800,000
amazon s3	483,000
amazon unbox	5,860,000
amazon.ca	64,000,000
amazon music	14,600,000
amazon.de	103,000,000
amazon .com	355,000,000

Table 1: Google Suggestions for *Amazon*

[1] This data was taken from Google ([8], [9]) via the web API on May 31, 2008.

[2] Google lists 209,000 results for this query and actually suggests it as one of the possible refinements for the search *Amazon Women.*

relevance feedback, adding terms from known relevant documents to a user's query. This technique is used in the TREC with query term weighting to improve the effective of users queries. Experiments with the TREC system show linear improvements in precision up to about 300 terms. [4].

Another classic expansion method, thesaurus lookup, expands a users query with synonyms from a pre-computed thesaurus . This technique, particularly if performed naively, can perform quite poorly. For example if, a query for *river bank* is expanded with financial terms, the results will be very poor.

Modern search engines all provide some form of query expansion. While the exact implementations of these algorithms are proprietary, examining the results [3] makes it clear that a combination of query popularity and spelling correction is used to determine which refinements to suggest.

The difference between our work and previous work is that the goal of these techniques is to increase precision. Our goal is to increase recall by eliminating redundancy in the automatic expansion terms.

3. DATA SET

Our experimental data was derived from a random selection of popular terms in one month of Yahoo! query logs. [4] The popular terms were then intersected with semantically clustered data from the Wikipedia corpus provided to us by Patrick Pantel ([15]).

We randomly selected 100 terms from this set that had at least two semantic meanings within the Wikipedia corpus. In the interest of civility, we manually filtered these terms for expletives and adult content. After filtering, we were left with a corpus of 64 words for analysis. For each of these terms we found approximately 1,000 popular refinements (where the term was the first word in a query string) from the Yahoo! search logs.

Our term set includes words like *java* (programming language/language, island) and *amazon* (source/website, river) that have a technology and non-technology meaning. It also includes the names of famous individuals *jennifer lopez* (artist, george clooney/brad pitt/ben affleck), *madonna* (artist, virgin) and items from television such as *mtv* (network, studio/distributor). In many cases, such as with *mtv* the semantic clusters of the term are closely related, in other cases such as *madonna* they are very divergent - representing cases where a word has two very different conceptual uses (Madonna the singer versus Madonna as a term for the Virgin Mary).

4. METHODOLOGY

We examined three different approaches, CBC Word Sense Similarity, Web Semantic Clustering, and Maximum Marginal Relevance Diversity to measure the similarity between candidate refinements and reduce the redundancy within the final set of search refinements.

[3]See the gSuggest.py file in our project submission to explore the refinements used by Google.

[4]Our agreement with Yahoo! Applied Research prohibits us from publishing specific features of the population; however, we can discuss the creation process with the course staff in more detail if required.

4.1 CBC Word Sense Similarity

Our initial approach to computing the similarity between candidate refinement terms was suggested by Patrick Pantel [16]. Leveraging Pantel's data from the Wikipedia corpus, we represented a refinement using a feature vector where each feature is composed via word co-occurance within a particular syntatic context. As an example [*sip* + VERB-OBJECT] is a feature for *wine*.

The first problem we found with this approach was sibling terms in hypernym or hyponym relations have very high similarity scores, even though they may prove to be valuable refinements from a search perspective. As an example, both *earth* and *jupiter* are offered as popular refinements for *planet* by Yahoo!, but would have high similarities using the word sense approach. [5]

Looking at the refinements for *weather*, in our console, you see both *toronto weather* and *chicago weather*. As the weighting toward diversity increases, *toronto* is eliminated from the refinement set very quickly due to it's similarity with *chicago*.

The final problem we found was a low overlap between the terms in the Wikipedia corpus and the queries that we collected from query logs. This led us to explore an alternate approach of computing similarity based on domain and url features drawn from Yahoo! query results. [6]

We attempted to use the word sense clusters from Pantel [16], but these clusters proved to broad to provide a good differentiation between candidate refinements. This led us to explore using web semantic similarity for clustering and differentiation.

4.2 Web Semantic Similarity Clustering

The first approach we took to semantically clustering the refinement terms is based on the observation that semantically related query terms usually share a similar set of results. For each refinement term, we fetched the 50 top-ranked query results from the Yahoo! search engine and constructed a binary feature vector based on the top query results.

We assigned two vectors to each refinement term. The first vector was based on URLs. The total feature set was all of the unique URLs returned by the query results. The vector for each term would have a 1 for each particular URL that its query results returned. The second vector was constructed similarly with unique hostname as a feature.

We calculated two separate k-means clusterings on the search terms, one for URLs and one for hostnames, using Euclidean distance for our distance function. For both types of feature vectors, we clustered the data into 20 different clusters. Because our data set was very sparse, we randomly selected 20 points out of the data set to be our initial centroids and a maximum of 100 iterations of the k-means algorithm were performed, to generate our clusters.

As we see in table 2, the domain clustering provides a very good approximation of lexical semantic clusterings and is much simpler to compute. Each of our example clusters captures a single meaning from the Wikipedia corpus [16].

[5]Etzioni [7] suggests several possible techniques that could be used to overcome this problem.

[6]For completeness, we include the word sense data in our project submission and have implemented a mode in the web console where you can use the word sense similarities to compute diversity.

Cluster 1478	Cluster 592
java island	javascript menu
java indonesia	cross browser javascript menus
java island map	javascript dropdown menu
west java	javascript menus
	this webshop because javascript

Table 2: Example Clusters for *Java* refinements

This approach is not without potential problems. Depending on how the search engine works, it can be difficult to differentiate terms like *java* from *java script*. Although, outside of the audience of technically sophisticated editors, it's unclear that the majority of human editors would distinguish between the two, either. Pantel [16] assigns the same lexical semantics (PROGRAMMING LANGUAGE/LANGUAGE) to both terms, so it is unclear if this is an issue across the broader population of users.

For almost all of the terms, there were outlying data points that did not fit well into the final set of clusters. One potential optimization is to investigate algorithms to filter out some of these outliers or group them into a single outliers cluster. Another optimization is to look at other possible distance metrics, which we discuss in the future work section.

Unfortunately, the clustering is very expensive to compute and we found that we could get a good approximation of the clustering results by using the domain and URL features directly.

4.3 MMR Diversity

To compute diversity we follow an approach similar to the Maximal Marginal Relevance approach of Carbonell [5]. We pick refinements by optimizing the MMR using a greedy discounting algorithm. For our measure of mariginal relevance we use the formula:

$$\begin{aligned} MMR &= argmax_x(MR(x, y, X)) \\ MR(x, y, X) &= (1 - \lambda) \log r(x, y) + \lambda\beta \log d(x, X) \end{aligned}$$

where x is a candidate refinement, y is the user's original search term, and X is the set of previously chosen candidate refinements. β is a hand-tuned scaling factor used to ensure that the magnitude of the diversity function $d(x, X)$ is roughly the same as the relevance function $r(x, y)$.

The relevance function for our marginal relevance equation $r(x, y)$ is defined as:

$$r(x, y) = \frac{c(x)}{c(y)}$$

where the $c(x)$ is the count of the occurrence of x as a search query from the search logs during our sampling period.

The diversity function $d(x, X)$ for our marginal relevance equation is defined as:

$$d(x, X) = \gamma d_{URL}(x, X) + (1 - \gamma)d_{DOMAIN}(x, X)$$

and γ is a weighting between the domain and url diversity terms. For our experiments we used a value of $\gamma = 0.7$. While tuning the function, we perceived that URLs have a higher precision for indicating diversity but are very sparse. Domains on the other hand had higher overlap, but less selectivity as far as diversity.

Given a function $S(x)$ which maps x to a set of up to 50 of the top URL results for the term x as determined by the Yahoo! search engine, we define our URL diversity function d_{URL} as follows:

$$\begin{aligned} d_{URL}(x, X) &= \frac{|\{u_i \mid u_i \in S(x) \wedge u_i \notin S(X)\}|}{max(|S(x)|, 25)} \\ S(X) &= \cup_{x' \in X} S(x') \end{aligned}$$

Similarly, using a function $D(x)$ which maps x to a set of the unique domains contained in the set $S(x)$, we compute the domain diversity function d_{DOMAIN} using the formula:

$$\begin{aligned} d_{DOMAIN}(x, X) &= \frac{|\{d_i \mid d_i \in D(x) \wedge d_i \notin D(X)\}|}{max(|D(x)|, 25)} \\ D(X) &= \cup_{x' \in X} D(x') \end{aligned}$$

5. IMPLEMENTATION

Our data processing system is implemented in a combination of Unix shell scripts, Python, and proprietary Yahoo! data processing system including a Hadoop cluster. The extracted data is batch processed and the results are loaded into a MySQL database. The web interface into the system was implemented using a combination of Python and PHP.

A proprietary data processing system is used to extract potential search terms from the query logs and extract potential candidate terms into a text file, then is properly anonymized. Once the potential term set is computed, the set of potential refinements is then extracted from the logs as well. Because this part of the system has to be run at Yahoo! , we designed it to be loosely coupled, allowing us to run the bulk of the system in a non-proprietary environment.

In Figure 1, we see the overall architecture of the system. The extracted query logs are processed to compute the intersection with the Wikipedia corpus [16] as described in the data set section, producing our final data set.

Once the data set of terms is determined, we use a Python script to query the Yahoo! search engine to get the top 50 results for each term and each refinement. The URL data is binned into unique host names, and both sets of data are loaded into a MySQL database. Clustering is performed and the pre-generated clusters are loaded into the database.

A web console, implemented in PHP, is used to display a search term along with its suggested refinements. A slider is provided to experiment with adjusting the weighting between diversity and popularity used in selecting a set of refinements, as the weighting changes the system dynamically recomputes the new set of refinements.

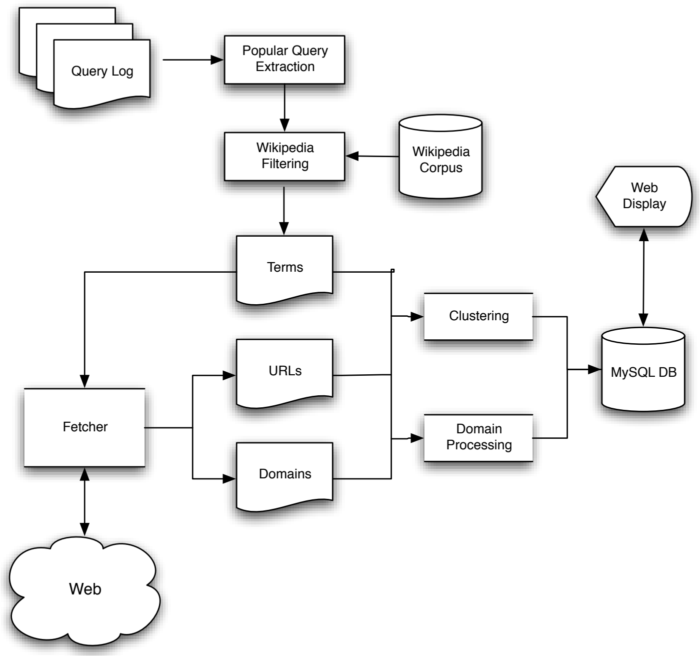

Figure 1: System Architecture

6. EVALUATION

6.1 Methodology

The purpose of eliminating redundancy is to create better refinement sets for the end user. Determining what is a better set of refinements is an inherently subjective judgement, so we set up an AB test to perform an extrinsic evaluation of our work. We pre-computed three different refinements sets for each of our terms with the diversity weighting set to $\lambda = 0$, $\lambda = 0.33$, and $\lambda = 0.80$. The pre-computed sets are loaded into a database for use in an editorial test.

We built a web application in PHP to execute the AB tests. When a test subject accessed the site, they were presented with two columns of ten search refinements as we see in Figure 2. One of our terms is randomly presented to the test subject with two sets of refinements. One of the refinements has a diversity weighting of 0, the other refinement set is randomly drawn from one of the positive diversity sets.

The test subjects are then asked to evaluate which set of refinements is better taking into the importance of the terms and what terms are already in the set. Subjects respond on a 5 point scale: A definitely better, A slightly better, indistinguishable, B slightly better, B definitely better. Subjects also are allowed to skip a test if they are unable to come to any judgement.

We recruited 17 volunteers from friends and acquaintances and asked them to execute the test. The subjects were roughly equally divided between males and females and were largely professionals in the 20 - 45 age group. There was a significant bias toward people employed in software development. [7] Over half of the subjects were native English speakers, and the remaining half speak a variety of other languages as their first language.

[7]While we acknowledge that this is not an ideal population from a sampling perspective; however, we do think it provides valuable human evaluation for the final project.

6.2 Methodological Problems

After the editorial testing commenced, it became clear that there were flaws in the way in which our samples were being presented. First, after answering questions from some confused text subjects, it was clear the we needed a better set of evaluation criteria to guide our human editors.

Second, partitioning the sets into shared and unshared terms seemed problematic. We originally did not make any distinction between the shared and unshared terms, but felt that it would be easier to do the comparison of sets if we did differentiate between the two types. This turned out to have a negative effect on the reviewers. The UI drew the test subjects attention toward the different sets and the subjects focused predominantly on that and not the refinement sets as a whole.

Finally, we needed to do better filtering on the data to remove adult, controversial, or offensive terms. One of the refinements derived from query logs was an unflattering description of a presidential candidate supported by one of the test subjects. The test subject fixated on the inappropriateness of this particular refinement and not the evaluation task. This proved to be a larger problem than we anticipated given our particular set of subjects.

6.3 Results

Despite the issues with the testing, we still wish to present the results of the experiment. While not conclusive, it does shed some light on problems that need to be addressed with the diversity model.

When our reviewers compared the results of the query refinements with no diversity to a diversity value $\lambda = 0.33$, the overall utility is about equal. Assigning a value of 1 to

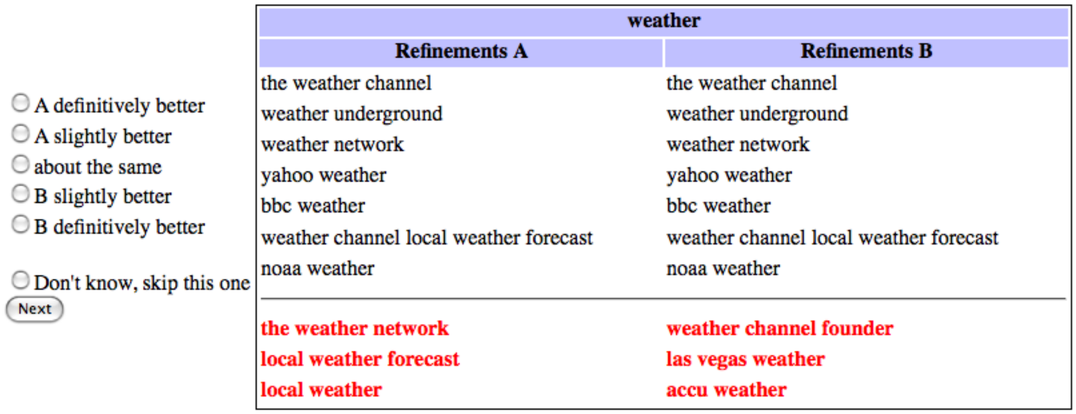

Figure 2: Editorial Test Screen

5 for each of the choices, starting with the less diverse being definitely better as 1 and ending with the more diverse being definitely better as 5, the median choice is 5. The mean is 2.946, the standard devivation is 1.207, and the mode was 4.

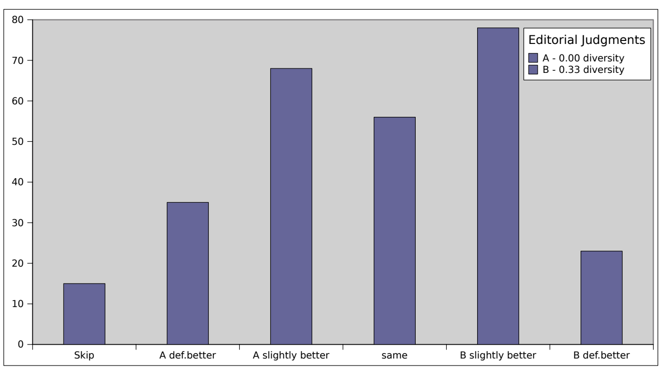

Figure 3: Diversity = 0.33 Test

When the diversity value was increased to $\lambda = 0.80$, the overall utility of the refinements presented was judged to be lower than no diveristy at all, as we see in Figure 4. Using the same 5 point scale as above, the median choice for this trial was 2, the mean 2.656, the standard deviation was 1.285, and the mode was 2.

6.4 Analysis

One problem with using our diversity metric to compute refinements set is false diversity. Terms favored by adult websites and spam websites show an artificially high diversity score when using url or domain based diversity scores. A common technique used by these type of web sites is to create a large number of different domains with little content to try and game linked-based analysis techniques.

A possible solution to this problem would be to apply spam and adult classification models to the URLs and domain prior to clustering to filter out the unwanted data. Since popular terms are often "attacked" by spammers, this could have the effect of filtering out some desirable terms. A second possible solution, is to use the classification as a parameter into the weighting function of the model instead of using linear weightings.

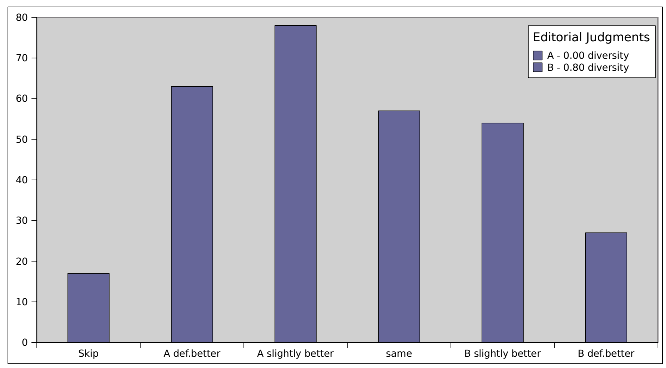

Figure 4: Diversity = 0.80 Test

The length of a refinement term affected how our test subjects rated the refinement set. The subjects penalized sets that contained long refinement strings like we see in the example for *ikea* in table 3. For this particular set, almost all of the test subjects rated the $\lambda = 0.0$ refinement set as definitely better than the $\lambda = 0.80$ set.

$\lambda = 0.0$	$\lambda = 0.80$
ikea store	ikea store
ikea furniture	ikea canada
ikea usa	ikea.no
ikea canada	ikea malaysia catalogue 2008
ikea uk	ikea milton keynes
ikea store locations	ikea west chester
ikea stores	ikea hacker
ikea usa furniture kitchen	ikea office furniture desks
ikea furniture store	ikea gutschein wartet
ikea malaysia	ikeagoddess

Table 3: *ikea* refinements

This data is confirmed by talking with some of the subjects after running the test. When we asked the subjects what kind of things they found made for a bad results, the length of the term was brought up as one of the problems. [8] Higher λ values had a tendency to introduce longer refinements into the candidate set, to address this problem some form of length penalty term could be added to a refinement selection model.

Test subjects brought up foreign language terms as another one of the problems with some of the refinement sets. Again, higher λ values increased the likelihood that foreign language terms would be introduced into the refinement set. Even when reviewers understood both languages, they told us that they did not like the mixed language sets. This issue could be addressed with either a dictionary lookup or language identification algorithm.

In other cases, using diversity to compute the refinement terms, demonstrated a real value. For the search term *yellow pages*, our test subjects universally preferred the high diversity set. The two sets are shown in table 4.

$\lambda = 0.0$	$\lambda = 0.80$
yahoo yellow pages	yahoo yellow pages
huns yellow pages	huns yellow pages
the huns yellow pages	yellow pages australia
the hunn's yellow pages	sbc yellow pages
the hun's yellow pages	the hunn s yellow pages
the hun yellow pages	verizon yellow pages online
the hunn s yellow pages	the hun yellow pages
yellow pages australia	cincinnati bell yellow pages
hun's yellow pages	the hunss yellow pages
sbc yellow pages	india yellow pages

Table 4: *yellow pages* **refinements**

Test subjects felt that the addition of terms like *india*, *verizon*, and *cincinnati* made this set of refinements much more useful than a large number of refinements for *huns*. In general, however, redundancy was less important to users for larger refinements sets.

7. CONCLUSIONS

The results of our user study indicate that applying a diversity metric to a set of search term refinements had a strong affect on the perceived value of the refinement set by a human reviewer. In some cases the addition of diversity improved the utility of the refinement set, while in others it clearly decreased the value of the set.

Naively applying a diversity metric to a set of refinement clusters does not work. In many cases, other factors, such as the language of a refinement term and the length of a refinement term negatively come into play and need to be accounted for in the selection. Also, particularly in the case of internet search refinements, controversial, adult, and spam need to be addressed in some fashion.

Clearly, the weighting between diversity and popularity is a factor. High weightings of diversity (like our $\lambda = 0.80$ trial) are not judged to be useful by end users, but lower weightings (around $\lambda = 0.30$) do show definite signs of value. However, for diversity to be effective, it needs to be included as part of a refinement selection model which includes factors for popularity, diversity, content classification, language and length.

8. FUTURE WORK

Our results are based on diversity clusters generated using a Euclidean distance function. Work by Lee [11] indicates that distances formulas involving squared terms (Euclidean/L2, Cosine) do not perform as well as distance formulas which do not use squared terms (Manhattan/L1, Jensen-Shannon, etc.). Recomputing our clusters with one of the better performing distance measures [9] is a simple experiment to evaluate a higher refinement

Because the value of the refinement diversity is highly subjective, we would explore changing several aspects of the AB editorial test. It is difficult for an editor to do a good job comparing two sets of ten refinements, adjusting the size of the refinement set to five refinements should increase the inter-judgement agreement and increase the reliability of the judgements. After refining the system through AB testing, we would further evaluate the technique using a real world bucket test and measure the value of refinements via clicks.

9. ACKNOWLEDGEMENTS

The authors would like to thank Patrick Pantel for his assistance and advice with this project. We would also like to thank Yahoo! Applied Research for granting us permission to use the anonymized query data for our report.

10. REFERENCES

[1] Anick, P. Using terminological feedback for web search refinement: a log-based study. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (New York, NY, USA, 2003), ACM, pp. 88–95.

[2] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] Bollegala, D., Matsuo, Y., and Ishizuka, M. Measuring semantic similarity between words using web search engines. In *WWW '07: Proceedings of the 16th international conference on World Wide Web* (New York, NY, USA, 2007), ACM, pp. 757–766.

[4] Buckley, C., Salton, G., Allan, J., and Singhal, A. Automatic query expansion using SMART: TREC 3. In *Text REtrieval Conference* (1994), pp. 0–.

[5] Carbonell, J., and Goldstein, J. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1998), ACM, pp. 335–336.

[8] One of our test subjects commented, "These terms are just too long."

[9] Given the size of the data set we are using and the computational resources available, it takes 30 hours to cluster the refinements by domains and 120 hours to cluster by URLs. We discovered this paper in the last week of the project, which did not leave enough time to take advantage of these results.

[6] Dagan, I., Lee, L., and Pereira, F. Similarity-based models of cooccurrence probabilities. *Machine Learning 34*, 1-3 (1999), 43–69.

[7] Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell. 165*, 1 (2005), 91–134.

[8] Google. Google search. http://www.google.com, 2008.

[9] Google. Google suggest. http://www.google.com/webhp?esrch=GoogleSuggestBeta, 2008.

[10] Haveliwala, T. H., Gionis, A., Klein, D., and Indyk, P. Evaluating strategies for similarity search on the web. In *WWW '02: Proceedings of the 11th international conference on World Wide Web* (New York, NY, USA, 2002), ACM, pp. 432–442.

[11] Lee, L. On the effectiveness of the skew divergence for statistical language analysis. In *Artificial Intelligence and Statistics 2001* (2001), pp. 65–72.

[12] Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval (Draft).* Cambridge University Press, 2008.

[13] Mitra, M., Singhal, A., and Buckley, C. Improving automatic query expansion. In *Research and Development in Information Retrieval* (1998), pp. 206–214.

[14] Pan, F., Roberts, A., McMillan, L., Threadgill, D., and Wang, W. Sample selection for maximal diversity. In *ICDM* (2007), IEEE Computer Society, pp. 262–271.

[15] Pantel, P. Lexical semantics demo. http://demo.patrickpantel.com/Content/LexSem/cbc.htm, 2008.

[16] Pantel, P., and Lin, D. Discovering word senses from text, 2002.

[17] Sahami, M., and Heilman, T. D. A web-based kernel function for measuring the similarity of short text snippets. In *WWW '06: Proceedings of the 15th international conference on World Wide Web* (New York, NY, USA, 2006), ACM, pp. 377–386.

[18] Segaran, T. *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. O'Reilly Media, Inc., August 2007.

[19] Xu, J., and Croft, W. B. Query expansion using local and global document analysis. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1996), ACM, pp. 4–11.